Method Families Concept: Application to Decision-Making Methods

Elena Kornyshova, Rebecca Deneckere, Colette Rolland

Method Families Concept: Application to Decision-Making Methods

Elena Kornyshova, Rébecca Deneckère, Colette Rolland

Centre de Recherche en Informatique, Université Paris 1 Panthéon-Sorbonne,
90 rue de Tolbiac, 75013 Paris, France
{kornyshova, denecker, rolland}@univ-paris1.fr

Abstract. The role of variability in Software engineering grows increasingly as it allows developing solutions that can be easily adapted to a specific context and reusing existing knowledge. In order to deal with variability in the method engineering (ME) domain, we suggest applying the notion of method families. Method components are organized as a method family, which is configured in the given situation into a method line. In this paper, we motivate the concept of method families by comparing the existing approaches of ME. We detail then the concept of method families and illustrate it with a family of decision-making (DM) method that we call MADISE.

Keywords: Situational Method Engineering, Method Family, Method Line, Decision-Making Methods.

1 Introduction

An information system development methodology is a set of ideas, approaches, techniques and tools used to transform organizational needs into an appropriate information system (IS). There are many and various application domains for these methodologies. However, because of this diversity, it is now clear that a universal method that could be applied to handle completely any IS development project does not exist. Method Engineering (ME) is a discipline which aims to bring effective solutions to the construction, improvement and modification of methods. Several authors tried to conceive methods that would be as effective and as adapted as possible to the IS development [1] [2]. However, this objective was not always reached, especially because the methods were not really adapted to projects situations. The situational methods were designed to correct this drawback. Situational Method Engineering (SME) finds its justification in the practical field analysis which shows that a method is never followed literally [3] [4]. It promotes the idea of using combined method parts, instead of complete methodologies, to specific situations [5].

However, these approaches are not widely spread in the practitioners' world. The components composition is a quite complicated process as there may exist some overlapping between concepts (which is the rationale behind the integration process of the assembly technique) and no SME approach really offers a simple and easily understandable way to construct a situational method. Furthermore, it is nowadays

acknowledged that contingency factors change continuously during the project life cycle imposing a continuous change management. This last feature raises the problem of the dynamic adaptation of methods, which has not been considered by current SME approaches [6].

Our proposition suggests to move away from this construction of methods 'on the fly' to the management of a set of similar components as a whole. Our proposal is to organize these components in method families to manage variability and commonalities in order to promote the reuse and the adaptability of method families. The method family is then configured in a given project in order to obtain an appropriate method line. We think that method families do exist today in companies and could beneficially be handled in an easier manner.

This paper is organized as follows. In Section 2, we present a brief state-of-the-art of the seven most known SME approaches in order to identify their drawbacks. We offer a general vision of the method family concept and describe its model in Section 3. We illustrate our proposal with the example of MADISE, a family of decision-making methods in Section 4. We conclude in Section 5.

2 SME Approaches: State-of-the-Art

In this section, we present the comparison of the seven main existing SME approaches and we give an analysis of their drawbacks.

The four views framework has proved its efficiency in enhancing the understanding of various engineering disciplines such as information systems engineering, requirements engineering, IS development process engineering and method engineering. Our point of view is that this framework concept can be used to help in understanding and comparing different SME approaches.

For our purpose, we define the SME four-dimensional framework as:

- **Objective view.** In the usage view, the corresponding dimension allows investigating the rationale of SME approaches.
- **Subject view.** This view expresses the dimension which deals with the representation of SME approaches, their nature.
- **Development view.** The development view deals with the process of constructing the SME approaches.
- **Usage view.** This view deals with different aspects that describe the SME approaches usage.

We propose a review of seven SME approaches. We choose our method panel in the set of the most widespread approaches and with the intention to offer a more complete study of the different views: Method fragment approach [7] [8] [9], Method Chunk Approach [3] [11], Method Configuration Approach (Component) [12] [13] [14], OPEN Process Framework [15] [16], Method Service Approach [17], Method Extension Approach [18], and FIPA (Foundation for Intelligent Physical Agent) approach [19] [20] [21] [22]. These seven approaches are compared according to the suggested framework. Table 1 resumes the results of this comparison (The detailed description of the SME approaches comparison can be found in [23]).

Table 1. SME Approaches' Review according to the Framework.

View	Attribute	Value	Method fragment	Method chunk approach	Method configuration approach	OPF approach	SO2M, method service approach	Method extension approach	FIPA approach
Objective	Covered way	{Way of thinking, Way of modeling, Way of organising, Way of supporting}	{Way of thinking, Way of modeling, Way of organising}	{Way of thinking, Way of modeling, Way of organising}	{Way of thinking, Way of modeling, Way of organising}	{Way of thinking, Way of modeling, Way of organising}	{Way of thinking, Way of modeling, Way of organising}	{Way of thinking, Way of modeling, Way of organising}	{Way of thinking, Way of modeling, Way of organising}
	Target issues	{Variability, Intentionality, Context-Awareness, Reusability, Conflict Resolution}	{Context-Awareness, Reusability}	{Intentionality, Context-Awareness, Reusability}	{Intentionality, Context-Awareness, Reusability}	{Context-Awareness, Reusability}	{Intentionality, Context-Awareness, Reusability}	{Intentionality, Context-Awareness, Reusability}	{Intentionality, Context-Awareness, Reusability}
Subject	Actor representation	{Role, Producer, User}			{Role}	{Role, Producer}			{Role}
	Knowledge dependency	{Yes, No}	{Yes}	{Yes}	{Yes}	{Yes}	{Yes}	{Yes}	{Yes}
	Knowledge representation	{Fragment, Chunk, Component, OPF Fragment, Service, Pattern}	{Fragment}	{Chunck}	{Component}	{OPF fragment}	{Service}	{Pattern}	{Fragment}
	Variability representation	{Yes, No}	{No}	{No}	{No}	{No}	{No}	{No}	{No}
	Context representation	{Reuse frame, Interface, Contingency factor, Development situation}	{Contingency factor}	{Reuse frame, Interface}	{Development situation}		{Interface}	{Interface}	{Development situation}
	Abstraction Level	{Conceptual, Technical}	{Conceptual}	{Conceptual}	{Conceptual}	{Technical}	{Technical}	{Conceptual}	{Conceptual}
Development	Knowledge construction	{Ad-hoc, Formalised}		{Formalised}	{Formalised}	{Formalised}		{Formalised}	{Ad-hoc}
	Reengeneering process	{Decomposition, Assembly, Instantiation}	{Decomposition}	{Decomposition}	{Decomposition}	{Decomposition}	{Decomposition}	{Instantiation}	{Decomposition}
	Knowledge organisation	{Repository, Organisational process}	{Repository}	{Repository}	{Repository, Organisational process}	{Repository}	{Repository, Organisational process}	{Repository, Organisational process}	{Repository}
	Context specification	{Yes, No}	{No}	{No}	{No}	{No}	{No}	{No}	{No}
	Tool/ Implementation	{Product storage and manipulation, Process operating, Retrieval, Construction}	{Product storage and manipulation, Retrieval, Construction}	{Product storage and manipulation, Retrieval}	N/S	N/S	{Product storage and manipulation, Retrieval, Construction}		
Usage	Process model	{Activity-oriented, Product-oriented, Decision-oriented}	{Activity-oriented}	{Decision-oriented}	{Activity-oriented}	{Activity-oriented}	{Decision-oriented}	{Decision-oriented}	{Activity-oriented}
	Construction flexibility	{Selection and adaptation of a method, Modular construction of a method, Method selection in a multi-method}	{Modular construction of a method}	{Modular construction of a method}	{Selection and adaptation of a method, Modular construction of a method}	{Modular construction of a method}	{Modular construction of a method}	{Selection and adaptation of a method}	{Modular construction of a method}
	Construction technique	{Assembly, Instantiation, Extension, Reduction, Agile Construction}	{Assembly}	{Assembly, Extension}	{Assembly, Extension, reduction}	{Assembly, Instantiation, Agile construction}	{Assembly}	{Instantiation, Extension}	{Assembly}
	Context Usage	{Project characterisation, Component characterisation, matching component with situation}			{Project characterisation}	{matching component with situation}	{Project characterisation, Component characterisation, matching component with situation}	{Component characterisation, matching component with situation}	{Component characterisation}

The framework analysis allows identifying the following main drawbacks of the studied SME approaches:

- **Approaches interoperability.** Despite some standardisation efforts of the ME community, all approaches are strongly coupled with their own notion of method component so the techniques developed in one approach are not usable in another one.
- **Component retrieval.** As there is no common interface between components, their retrieval is made dependent of their nature. Locating the needed component may require searching several repositories rather than having them in one centrally available place [6]. Moreover, most of the approaches don't propose an organisational process to handle the components, which complicate their retrieval in the execution of the construction process.
- **Variability.** The variability issue is not taken into account in any approach so there is no representation of the common or variables parts in the current approaches.
- **Contextualization.** In order to use the context variability in an optimistic way, it is necessary to have the three parts of the context usage: the project characterisation (to describe the development situation, which evolves continuously), the component characterisation (to describe the reuse context) and the matching between them (to be able to choose the appropriate component in the appropriate situation). Only the method service approach proposes this contextualisation. In addition, none of the existing SME approaches does suggest a methodology for defining the context of methods.

3 Method Family Concept

This section offers the method family model, the general vision of the method family construction and usage, and the method family organization.

3.1 Method Family Model

The concept of method family is described with the meta-model of Figure 1 (using the UML formalism).

The ***method component*** concept has the same semantic that the classic SME methods (a building block, subset of a method), which may contains other components.

We propose in this work the concept of ***method family***, which is a set of several organized method components for a specific domain.

In a method family, some components may be considered either as ***common*** or ***variable*** as each situation may require specific components. These situations, where it is necessary to choose between several components are called ***variation point***. The relationship between the variation point and the component defines the ***variability dependency,*** which can be ***mandatory*** or ***optional*** (and respectively corresponds to the common and variable method components). A set of variable method components represent the alternatives offered to the engineer at a specific variation point, it is a way to realize variability. This representation of variability was inspired from [24] which discusses the variability of software product lines.

The ***method component context*** must be specified. It characterizes all possible situations in which this component may be applied. The ***project context*** includes all characteristics of the situation at hand, which matches some variable method component contexts. These two context concepts are specified on the same basis as they inherit from the ***context***. This allows a better matching between the situation and the components for the configuration.

A variable method component is selected following the situation at hand (i.e. the project context). The set of these ***selected method components***, together with the common method components, represents a ***project method line***.

A ***method line*** is either an ***initial method*** (a method already known) or a ***project method line*** (defined with our process). The project method line includes the mandatory components and those selected for the given project based on the project characteristics. A method family is composed of a set of method lines. This concept allows regrouping several method lines for a specific domain. Each method line or method family may itself be considered as a method component.

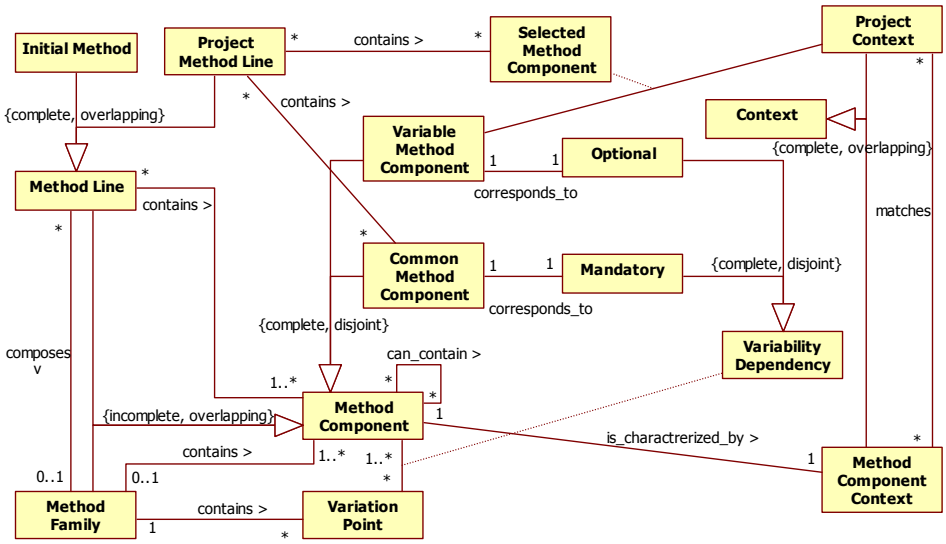

Fig. 1. Method Family Meta-Model.

Based on the comparison framework, the method family is characterized as follows:

```
Objective View
   Covered way = {Way of thinking, Way of modeling, Way of
      organizing}
   Target issues = {Variability, Intentionality, Context-Awareness,
      Reusability, Conflict Resolution}
Subject View
   Actor representation = {Role, User}
   Knowledge dependency = {No}
   Knowledge representation = {Fragment, Chunk, Component, OPF
      Fragment, Service, Pattern}
```

```
   Variability representation = {Commonalities and variability}
   Context representation = {Method service context, Contingency
      factor, Development situation}
   Abstraction Level = {Conceptual}
Development View
   Knowledge construction = {Formalized}
   Reengineering process = {Decomposition, Assembly}
   Knowledge organization = {Repository, Organizational process}
   Context specification = {yes}
   Tool/ Implementation = {Product storage and manipulation, Process
      operating, Retrieval, Construction}
Usage View
   Process model = {Decision-oriented}
   Construction flexibility = {Method selection in a multi-method}
   Construction technique = {Agile construction}
   Context Usage = {Project characterization, Component
      characterization, matching component with situation}
```

Thus, we can consider that the concept of method family allows overcoming the established drawbacks:

- This approach is independent of the knowledge representation as the concept of family may be applied to any component type (fragment, chunk, component and so on).
- A method family organizes components of the same domain in a way which enables their easier usage as it is already a subset of the potential method components. Then, the method family is configured according to a project needs. It allows avoiding the retrieval step (in several method bases) and composition step (as the method family is already composed of the method components).
- A method family is based on the separation of common and variable components in each variation point. This facilitates the use of SME approaches in practice.
- The given approach covers all necessary elements for the contextualization as it handles project characterization, component characterization, and matching component with situation.

3.2 General Vision of Method Family Construction and Usage

A method family is constructed based on the existing methodologies in a given field and for their further usage in different projects. Figure 2. gives an overview of method families and their usage.

The first step is to construct the method family from the existing methods. The next step is to configure the method family in order to obtain a method line adapted to the specific conditions of the project at hand. Finally, the obtained method line is applied in this project. On this basis, we define three following processes:

- Method family definition process;
- Method line configuration process;
- Method line application process.

Method family definition process. This process allows constructing method family from existing methods. Existing methods are decomposed into modular components

(like in different SME approaches) and are combined into the same model (i.e. a *family*). The combination of method components follows two main principles:

- Identification of variation points, common and variable components;
- Identification of components having the same goal and the same target product, but different ways of working.

At this step, the variability is taken into account as common and variable parts are shown. Alternative method components are identified and organized as variants in a given variation point. The difference with the usual SME approaches is that we combine all the components of the same domain in order to obtain a more 'generic' method (the family).

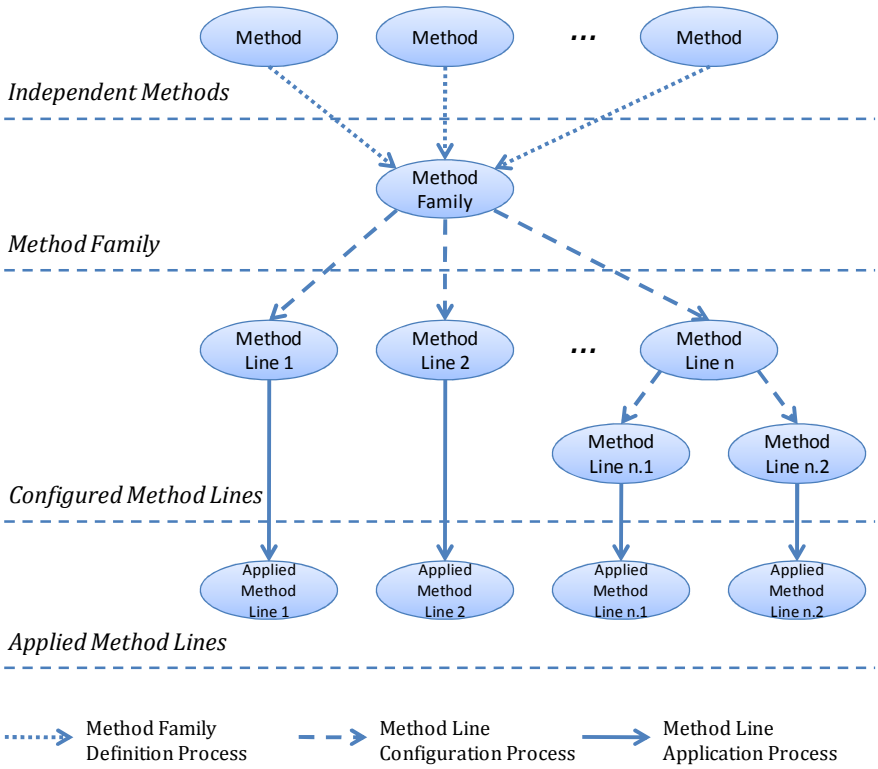

Fig. 2. General Vision of Method Families.

Method line configuration process. A method line is obtained following the configuration of the family according to various criteria.

We suggest three kinds of configuration:

- ***Complete method line selection.*** This configuration type helps to select between all the possible method lines.
- ***Method components sub-set selection.*** This kind of configuration allows to select a sub-set of components based on the context characteristics in a given project.

- ***Step by step method components selection.*** In this configuration type, the method line is configured during the project realization as the components are selected one by one.

Variation points facilitate the configuration process as common and variable components and the definition of their context enable to configure the method family according to the project needs. These kinds of configurations were described in [25]. It is also possible to run multiple sequential configurations to acquire the desired method line.

Method line application process. Once the method line has been created, a final configuration can be applied to obtain the method that will be used in a specific case.

The method family usage aims at constructing a method 'on the fly', following the project characteristics. However, as usual SME approaches deal with the construction process itself and its difficulties (integration *versus* association, for instance), method families offers a way to simplify the work of practitioners with a decomposition of the construction process. The method engineer constructs the method family but the practitioner just has to configure the method family to obtain a specific method adapted to his needs. The construction of the large method family base (repository) is justified by the need to provide a more flexible and context-aware usage of methods belonging to the same domain.

3.3 Method Family Organization

We use the MAP formalism [26] for representing method family and for organizing method components within method families.

A map is presented as a graph where nodes are *intentions* and edges are *strategies*. The key concept of a Map is the notion of the section which is an aggregation of two specific intentions, the *source intention* and the *target intention*, linked together with a *strategy*. It embeds the knowledge corresponding to a particular method component to achieve an intention (the target intention) from a specific situation (the source intention) following a particular technique (the strategy). When dealing with method families modeled by maps, each method component is represented by a map section, as shown in Figure 3.

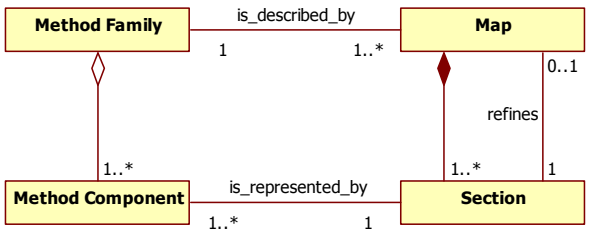

Fig. 3. Map and Method Family Correspondence.

This kind of organization provides the mean for ensuring the variability within method families.

4 Method Family Application to Decision-Making Methods

In this section, we show MADISE DM Method Family and two method lines obtained from this family.

4.1 MADISE Decision-Making Method Family

The MADISE DM Method Family describes the generic DM process including the main activities used for DM. As in the previous section, we have selected the Map formalism for representing the family of DM methods. The DM map is a collection of DM method components organized into a family in order to allow its further configuration according to a given situation. The MADISE DM Method Family modelled with MAP is presented in Figure 4.

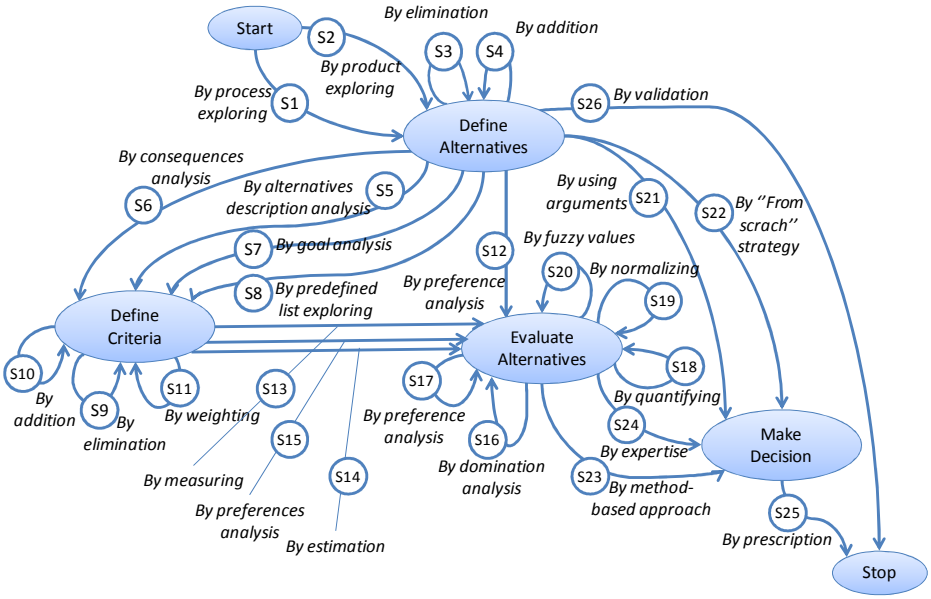

Fig. 4. MADISE DM Method Family (DM Map).

The usage of the DM Family is characterized by data which are required for starting and finishing the corresponding process. This implies the identification of the Input and Output data. Two kinds of information are required before beginning the use of the DM Map: the decision object and the decision problem which are *Input data*. The *Output data* is a decision made according to the identified decision

problem. The DM output could also have a NULL value if the decision is not made (for different reasons, such as a lack of information, not valid alternatives etc.). Figure 5. summarizes the Input and Output Data.

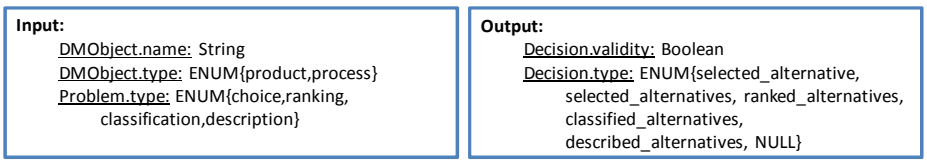

Fig. 5. Input and Output Data of the DM Method Family.

The DM Map contains four main intentions: Define Alternatives, Define Criteria, Evaluate Alternatives, and Make Decision.

The engineer starts the MADISE process by reaching the *Define Alternatives* intention. At this stage, an alternative set (or alternative family) is generated.

The *Define Criteria* intention is not mandatory. The engineer selects it if he wants to arbitrate between alternatives based on multiple factors. At this stage, a set of criteria for alternatives evaluation is defined, in particular only one criterion.

The *Evaluate Alternatives* intention aims at constructing the evaluation matrix (or decision matrix) [27].

At the *Make Decision* stage, a prescription for a decision is made.

4.2 Decision-Making Method Lines

In this section, our goal is to show that existing DM processes could be expressed through the MADISE DM method family. For doing this, each DM process must be represented as a MADISE line (i.e. a sub-set of MADISE sections). In order to illustrate this, we have chosen two existing and well-known DM processes: the cost-value approach for requirements prioritization [28], and tool selection from the Rational Unified Process (RUP) [29]. Their DM processes are captured and expressed as method lines in the following sub-sections.

Application Case: the Cost-Value Requirements Prioritization Approach. The cost-value requirements prioritization approach [28] aims at ranking requirements using the AHP DM method. The AHP (Analytic Hierarchy Process) proposed per T.L. Saaty [30]. As a shot reminder, this method is based on pair-wise comparison between alternatives and/or criteria and aggregation of comparison results into a quantitative indicator (score).

Figure 6. shows the DM method line corresponding to the cost-value approach.

The cost-value approach trajectory through the MADISE Map is as follows. The product based strategy is available for identifying candidate requirements (the *By product exploring* strategy is selected). This approach suggests reviewing candidate requirements for ensuring their completeness and correctness. Therefore, requirements can be added to or removed from the initial set (The *By elimination* and *By addition* strategies are selected). The approach defines two criteria describing requirements:

relative cost and relative value. These criteria are predefined by the cost-value approach (The *By predefined list exploring* strategy is selected). Actors (users and engineers) express their preferences by pair-wise comparison for defining the relative value and cost of candidate requirements (The *By preferences analysis* strategy is selected). The aggregated value obtained by AHP application is used for ranking requirement. The cost-value approach uses also a cost-value diagram in order to assist DM (The *By method-based approach* strategy is selected). A consistency index is calculated in order to check the result validity (The *By validation* strategy is selected). The DM components used by the cost-value approach are resumed in Table 2.

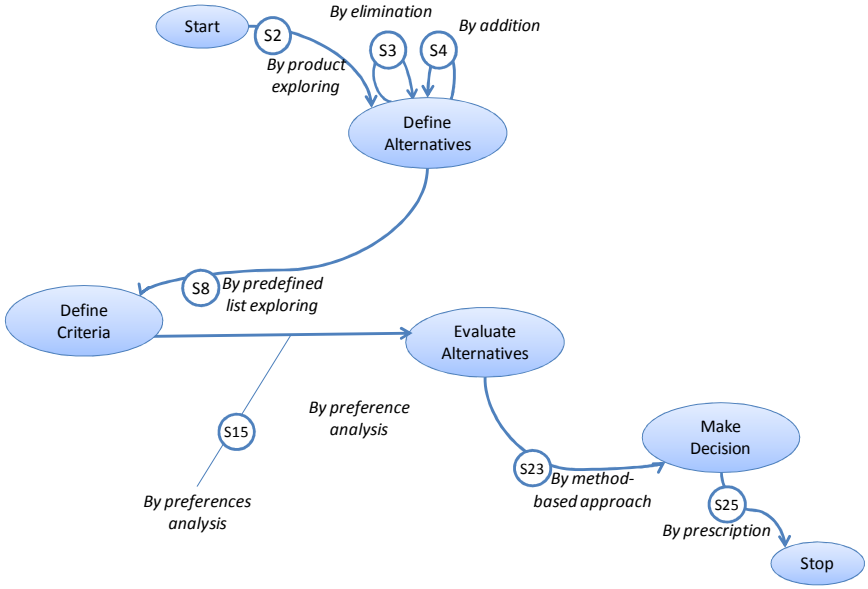

Fig. 6. DM Method Line of the Cost-Value Requirements Prioritization Approach.

Table 2. DM Method Line of the Cost-Value Approach: DM Components List.

Section	Component Name	Component Signature
S2	Define alternatives list by product exploring	<Start, By product exploring, Define Alternatives>
S3	Refine alternative list by elimination	<Define Alternatives, By elimination, Define Alternatives>
S4	Refine alternative list by addition	<Define Alternatives, By addition, Define Alternatives>
S8	Define criteria by predefined list exploring	<Define Alternatives, By predefined list exploring, Define Criteria>
S15	Evaluate alternatives by preferences analysis according to a criterion	<Define Criteria, By preferences analysis, Evaluate Alternatives>
S23	Make decision by method-based approach	<Evaluate Alternatives, By method-based approach, Make Decision>
S25	Prescribe decision	<Make Decision, By presctiption, Stop>

Figure 7. shows the input and output data for the DM method line corresponding to the cost-value approach. The DM object is *requirement* which is of the *product* type. The problem type is *ranking*. The output is the validated decision representing ranked requirements.

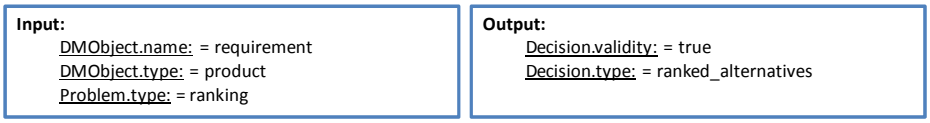

Fig. 7. Input and Output Data of the DM Method Line of the Cost-Value Requirements Prioritization Approach.

Application Case: the Tool Selection in RUP. The second example deals with tool selection and was taken from the RUP [29]. The RUP provides a wealth of guidance on software development practices. One of these practices is "Select and Acquire Tools". This task guides the selection of tools that fit project needs.

The DM method line representing the tool selection in RUP is illustrated at Figure 8.

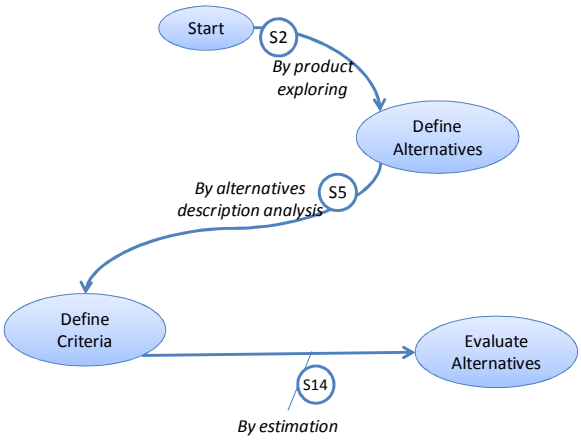

Fig. 8. DM Method Line of the Tool Selection in RUP.

The tool selection trajectory includes the following steps. A tool to select is considered as a product (The *By product exploring* strategy is selected). One of the steps in this task is to collect information about tools in order to decide which tool is suitable for the project at hand. The suggested criteria are: (i) tool criteria (features and functions, integration, applicability, extendibility, team support, usability, quality, performance, maturity); (ii) vendor criteria (stability, support availability, training, availability, growth direction); (iii) cost (acquisition cost, implementation cost, maintenance cost). These criteria are based on the tools description (The *By alternatives description analysis* strategy is selected). The RUP proposes to grade each criterion for evaluating candidate tools. The engineer estimates tools according

to different scales. Therefore, the evaluation is subjective (The *By estimation* strategy is selected). The recommendations of the RUP methodology stop at this stage. RUP does not contain any method for aggregating evaluations. Table 3. shows the set of the DM components retrieved in the RUP tool selection task.

Table 3. DM Method Line of the RUP Tool Selection: DM Components List.

Section	Component Name	Component Signature
S2	Define alternatives list by product exploring	<Start, By product exploring, Define Alternative>
S5	Define criteria by alternatives description analysis	<Define Alternative, By alternatives description analysis, Define Criteria>
S14	Evaluate alternatives by estimation	<Define Criteria, By estimation, Evaluate Alternatives>

Figure 9. illustrates the input and output data used in the RUP tool selection DM method line. The DM object is *tool,* which is a *product*. The goal is to select a tool. Therefore, the problem type is *choice*. The output is the validated decision, which corresponds to a selected tool.

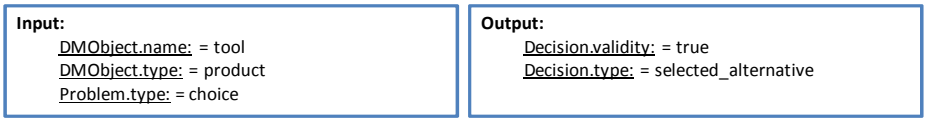

Fig. 9. Input and Output Data of the DM Method Line of the Tool Selection in RUP.

As we can see, the cost-value approach provides a more detailed guideline for DM. It allows a complete DM process from the alternatives definition to the decision validation. It contains a possibility to dynamically adjust an alternatives set. Two examples have different strategies for evaluating alternatives. The tool selection approach is simpler to carry out but it does not contain any decision-making and validation steps (we can see that there is no corresponding DM component in the DM method line).

Both examples are expressed as DM method lines as we have identified a suitable trajectory in the DM method family for each of them. The two DM processes are completely covered by the MADISE approach. Therefore, these examples help in validating the MADISE DM method family.

Moreover, another aspect is highlighted within these examples as the DM method family can contribute to improving existing DM models. For instance, the tool selection approach does not provide any advice for aggregating values, as mentioned above. Therefore, it can be completed by the DM component *Make decision by method-based approach* (the <Evaluate Alternatives, By method-based approach, Make Decision> section) in order to include an aggregation method. In the same way, the cost-value approach may be enhanced by the DM component *Discard alternatives by domination analysis* (the <Evaluate Alternatives, By domination analysis, Evaluate Alternatives> section) in order to eliminate dominated alternatives and, in this way, to simplify the AHP method application.

5 Conclusion

Following a brief state of the art of the main existing SME approaches, we have identified some drawbacks which may explain the reluctance of practitioners to use these SME approaches. We propose in this work a way to solve some of them with the notion of method family. We introduce the method family concept and we illustrate its application with the decision-making methods and method lines.

Method families help to organize a set of components for a specific domain. Thus, the engineer selects a specific method line, inside a family, to apply on its specific project. This approach allows using interoperable components, organized for a specific domain, based on the situation context.

Our future research includes validating this method family approach for all its steps: method family construction for several domains, method line configuration for different specific processes and method line application for several case studies. The MADISE Decision-making family is currently under evaluation by researchers and practitioners from differ fields.

6 References

1. Firesmith D., Henderson-Sellers B., The OPEN Process Framework. An Introduction, Addison-Wesley (2001)
2. Rolland C., Cauvet C., Object-Oriented Conceptual Modelling, CISMOD'92, International Conf. on Management of Data, Bangalore (1992)
3. Ralyte J. Method chunks engineering, PhD thesis, University of Paris 1-Sorbonne, (2001)
4. Mirbel I., De Rivieres V., Adapting Analysis and Design to Software Context : The jecko Approach, In 8th International Conference on Object Orirented Information Systems (2002)
5. Ralyte, J., Rolland, C., An approach for method reengineering. In proceedings of the 20th International Conference on Conceptual Modeling (ER'01), Yokohama, Japan, November 2001. H. Kunii, S. Jajodia, A. Solvberg (Eds.), LNCS 2224, Springer-Verlag (2001)
6. Rolland C., Method engineering: towards methods as services. Software Process: Improvement and Practice 14(3): 143-164 (2009)
7. Brinkkemper, S.: Method Engineering: engineering of information systems development method and tools, Information and Software Technology, 38(7), (1996)
8. Harmsen, A.F., Brinkkemper, J.N., Oei, J.L.H.: Situational Method Engineering for information Systems Project Approaches Int. IFIP WG8. 1 Conference in CRIS series: "Methods and associated Tools for the Information Systems Life Cycle" (A-55), North Holland (Pub.), 169-194 (1994)
9. Brinkkemper, S., Saeki, M., Harmsen, A.F.: A method engineering Language for the description of systems development methods, in proceedings of the conference CAISE 01. Springer Verlag. Interlaken, Switzerland, (2001)
10. Van Slooten K., Hodes B., Characterising IS development projects. In proceedings of the IFIP WG8.1 Conference on Method Engineering (1996)
11. Ralyté, J., Deneckere, R., Rolland, C.: Towards a Generic Model for Situational Method Engineering, in proceedings of the CAISE'03, Springer Verlag, Velden, Austria (2003)
12. Karlsson F., Agerfalk P.J., Method configuration: adapting to situational characteristics while creating reusable assets. Information and Software Technology Vol. 46 (9), (2004)
13. Wistrand, K., Karlsson, F.: Method components: Rationale revealed, in proceedings of CAISE 04, Springer-Verlag. Riga, Latvia (2004)

14. Agerfalk, P.J.: Information systems actability: Understanding Information Techology as a Tool for Business Action and Communication. Doctoral dissertation. Dept. of Computer and Information Science, Linköping University, (2003)
15. Henderson-Sellers, B.: Process meta-modelling and process construction: examples using the OPF. Ann. Software Engineering, 14(1-4), 341-362 (2002)
16. Henderson-Sellers B., Gonzalez-Perez C., McBride T., A meta-model for assessable software development methodologies. Software Quality Journal, 13(2) (2005)
17. Guzélian, G., Cauvet, C.: SO2M : Towards a Service-Oriented Approach for Method Engineering, in: the 2007 World Congress in Computer Science, Computer Engineering and Applied Computing, in the proceedings of the international conference IKE'07, Las Vegas, Nevada, USA (2007)
18. Deneckère R. Approche d'extension de méthodes fondée sur l'utilisation de composants génériques, PhD thesis (In French), University of Paris 1-Sorbonne (2001)
19. Cossentino M., Seidita V., M. Cossentino and V. Seidita, Composition of a new process to meet agile needs using method engineering. Software Engineering for Large Multi-Agent Systems Vol. III. LNCS Series, Vol. 3390. Springer-Verlag (2005)
20. Terracina G., Garro A., Ursino D., A multi-agent system for supporting the predition of protein structures. ICAE, 11(3) IOS Press, Amsterdam, Netherlands 256-280 (2004)
21. Method fragment definition, FIPA Document (accessed by November, 2003), http://www.fipa.org/activities/methodology.html (2003)
22. Cossentino M., Gaglio S., Henderson-Sellers B., Seidita V., A metamodelling approach for method fragment comparison, Proceedings of the 11th International Workshop on Exploring Modeling Methods in Systems Analysis and Design (EMMSAD), Luxembourg (2006)
23. Kornyshova E. and Deneckere R., A Framework for Comparing SME Approaches, Working paper, Centre de Recherche en Informatique, University of Paris 1 (2010)
24. Pohl K., Böckle G., Van der Linden F., Software Product Line Engineering: Foundations, Principles, and Techniques, Springer Verlag (2005)
25. Deneckere, R. and Kornyshova, E., Process line configuration : an indicator-based Guidance of the Intentional Model MAP. EMMSAD'10, Hammamet, Tunisia (2010)
26. Rolland, C., Prakash, N., Benjamen, A., A Multi-Model View of Process Modelling. Requirements Engineering. Volume 4, Number 4. Springer-Verlag London Ltd (1999)
27. Roy B., Multicriteria Methodology for Decision Aiding, Dordrecht, Kluwer Academic Publishers (1996)
28. Karlsson J., Ryan K.: A Cost–Value Approach for Prioritizing Requirements, IEEE Software (1997)
29. Rational Unified Process, Electronic Resource (accessed by June, 2007) http://www-306.ibm.com/software/awdtools/rup/ (2007)
30. Saaty T.L., The Analytic Hierarchy Process, NY, McGraw Hill (1980)